

# Machine Learning Overview and Neural Network

## Deep Learning (DSE316/616)

Vinod K Kurmi  
*Assistant Professor, DSE*

Indian Institute of Science Education and Research Bhopal

Aug 03, 2022



# Lecture overview

- What is learning?
- Types of machine learning problems
- Image classification
- Perceptron
- Multilayer Neural Network
- Disclaimer: Much of the material and slides for this lecture were borrowed from —Bernhard Schölkopf's MLSS 2017 lecture, —Tommi Jaakkola's 6.867 class, —Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class  
—Hongsheng Li's ELEG5491 class —CMP784: Deep Learning Fall 2021 Erkut Erdem Hacettepe University

# Lecture overview

- What is learning?
- Types of machine learning problems
  - Image classification
  - Perceptron
  - Multilayer Neural Network
- Disclaimer: Much of the material and slides for this lecture were borrowed from —Bernhard Schölkopf's MLSS 2017 lecture, —Tommi Jaakkola's 6.867 class, —Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class  
—Hongsheng Li's ELEG5491 class —CMP784: Deep Learning Fall 2021 Erkut Erdem Hacettepe University

# Lecture overview

- What is learning?
- Types of machine learning problems
- Image classification
- Perceptron
- Multilayer Neural Network
- Disclaimer: Much of the material and slides for this lecture were borrowed from —Bernhard Schölkopf's MLSS 2017 lecture, —Tommi Jaakkola's 6.867 class, —Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class  
—Hongsheng Li's ELEG5491 class —CMP784: Deep Learning Fall 2021 Erkut Erdem Hacettepe University

# Lecture overview

- What is learning?
- Types of machine learning problems
- Image classification
- Perceptron
- Multilayer Neural Network
- Disclaimer: Much of the material and slides for this lecture were borrowed from —Bernhard Schölkopf's MLSS 2017 lecture, —Tommi Jaakkola's 6.867 class, —Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class  
—Hongsheng Li's ELEG5491 class —CMP784: Deep Learning Fall 2021 Erkut Erdem Hacettepe University

# Lecture overview

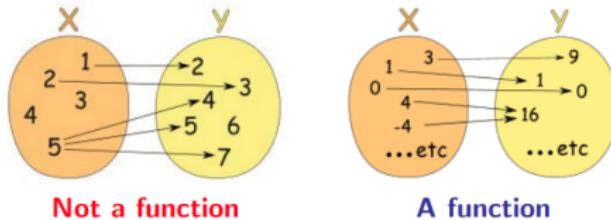
- What is learning?
  - Types of machine learning problems
  - Image classification
  - Perceptron
  - Multilayer Neural Network
- Disclaimer: Much of the material and slides for this lecture were borrowed from —Bernhard Schölkopf's MLSS 2017 lecture, —Tommi Jaakkola's 6.867 class, —Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class —Hongsheng Li's ELEG5491 class —CMP784: Deep Learning Fall 2021 Erkut Erdem Hacettepe University

# Lecture overview

- What is learning?
- Types of machine learning problems
- Image classification
- Perceptron
- Multilayer Neural Network
- Disclaimer: Much of the material and slides for this lecture were borrowed from —Bernhard Schölkopf's MLSS 2017 lecture, —Tommi Jaakkola's 6.867 class, —Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class  
—Hongsheng Li's ELEG5491 class —CMP784: Deep Learning Fall 2021 Erkut Erdem Hacettepe University

# Machine learning aims at learning a function

- Given an input value or vector, a function assigns it with a value or vector.
- “One-to-many” mapping is not a function. “Many-to-one” mapping is a function.



- Note that a function can have a vector output or matrix output. For instance, the following formula is still a function

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = f \left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} x_1 + x_2 \\ x_1 x_2 \end{bmatrix} \quad (1)$$

# Function estimation

- We are interested in predicting  $y$  from input  $x$  and assume there exists a function that describes the relationship between  $y$  and  $x$ , e.g.,  $y = f(x)$ .
- If the function  $f$ 's parametric form is fixed, prediction function  $f$  can be parametrized by a parameter vector  $\theta$
- Estimating  $f$  from a training set  
$$D = \{(x_1^{train}, y_1), (x_2^{train}, y_2), \dots (x_n^{train}, y_n)\}.$$
- With a better design of the parametric form of the function, the learner could achieve better performance.
- This design process typically involves domain knowledge.

# Function estimation

- We are interested in predicting  $y$  from input  $x$  and assume there exists a function that describes the relationship between  $y$  and  $x$ , e.g.,  $y = f(x)$ .
- If the function  $f$ 's parametric form is fixed, prediction function  $f$  can be parametrized by a parameter vector  $\theta$
- Estimating  $f$  from a training set  
 $D = \{(x_1^{train}, y_1), (x_2^{train}, y_2), \dots (x_n^{train}, y_n)\}$ .
- With a better design of the parametric form of the function, the learner could achieve better performance.
- This design process typically involves domain knowledge.

# Function estimation

- We are interested in predicting  $y$  from input  $x$  and assume there exists a function that describes the relationship between  $y$  and  $x$ , e.g.,  $y = f(x)$ .
- If the function  $f$ 's parametric form is fixed, prediction function  $f$  can be parametrized by a parameter vector  $\theta$
- Estimating  $f$  from a training set  
$$D = \{(x_1^{train}, y_1), (x_2^{train}, y_2), \dots (x_n^{train}, y_n)\}.$$
- With a better design of the parametric form of the function, the learner could achieve better performance.
- This design process typically involves domain knowledge.

# Function estimation

- We are interested in predicting  $y$  from input  $x$  and assume there exists a function that describes the relationship between  $y$  and  $x$ , e.g.,  $y = f(x)$ .
- If the function  $f$ 's parametric form is fixed, prediction function  $f$  can be parametrized by a parameter vector  $\theta$
- Estimating  $f$  from a training set  
$$D = \{(x_1^{train}, y_1), (x_2^{train}, y_2), \dots (x_n^{train}, y_n)\}.$$
- With a better design of the parametric form of the function, the learner could achieve better performance.
- This design process typically involves domain knowledge.

# Function estimation

- We are interested in predicting  $y$  from input  $x$  and assume there exists a function that describes the relationship between  $y$  and  $x$ , e.g.,  $y = f(x)$ .
- If the function  $f$ 's parametric form is fixed, prediction function  $f$  can be parametrized by a parameter vector  $\theta$
- Estimating  $f$  from a training set  
$$D = \{(x_1^{train}, y_1), (x_2^{train}, y_2), \dots (x_n^{train}, y_n)\}.$$
- With a better design of the parametric form of the function, the learner could achieve better performance.
- This design process typically involves domain knowledge.

# Types of machine learning problems

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning

# Types of machine learning problems

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning

# Types of machine learning problems

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning

# Types of machine learning problems

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning

# Image Classification: a core task in Computer Vision



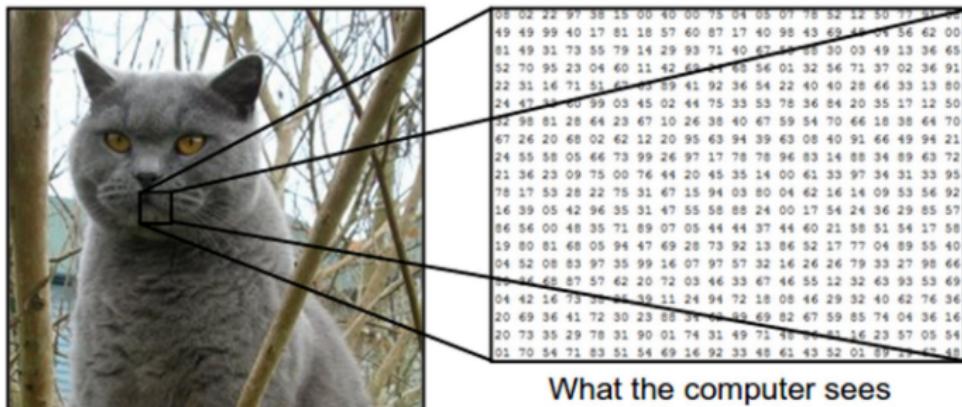
(assume given set of discrete labels)  
{dog, cat, truck, plane, ...}



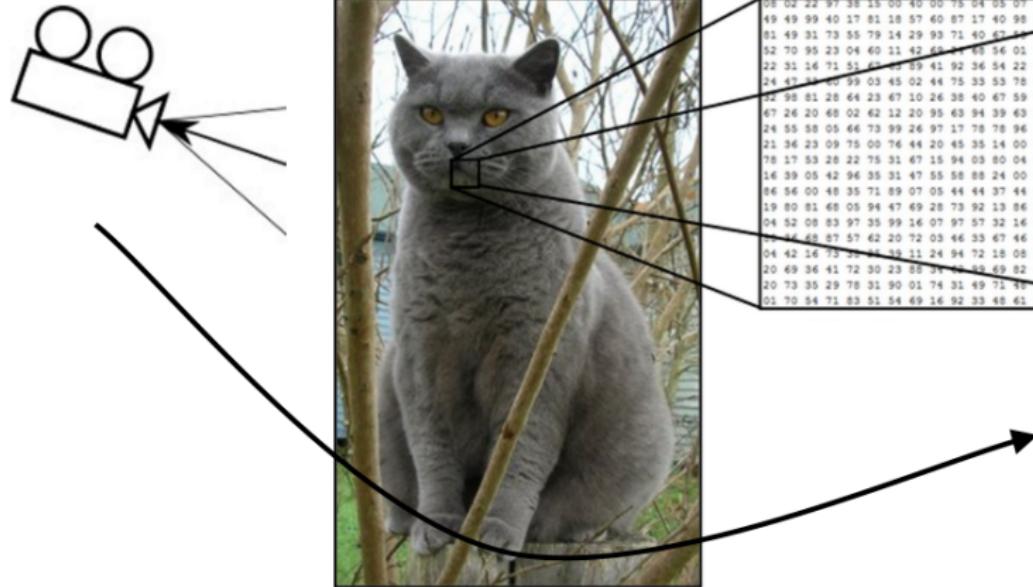
cat

# The problem: semantic gap

- Images are represented as 3D arrays of numbers, with integers between [0, 255].
- e.g.  $300 \times 100 \times 3$  (3 for 3 color channels RGB)



# Challenges: Viewpoint Variation



# Challenges: Illumination



# Challenges: Deformation



# Challenges: Occlusion



## Challenges: Background clutter



# Linear classifier: image classification



image parameters

$$f(\mathbf{x}, \mathbf{W})$$



**10** numbers,  
indicating class  
scores

**[32x32x3]**

array of numbers 0...1  
(3072 numbers total)

# Linear classifier: image classification

$$f(x, W) = Wx$$



**10** numbers,  
indicating class  
scores

**[32x32x3]**

array of numbers 0...1

# Linear classifier: image classification



[32x32x3]

array of numbers 0...1

$$f(x, W) = \boxed{W} \boxed{x}$$

**10x1**      **10x3072**

**3072x1**

**10** numbers,  
indicating class  
scores

parameters, or “weights”

# Linear classifier: image classification



[32x32x3]

array of numbers 0...1

$$f(x, W) = \boxed{W} \boxed{x}$$

**10x1**      **10x3072**

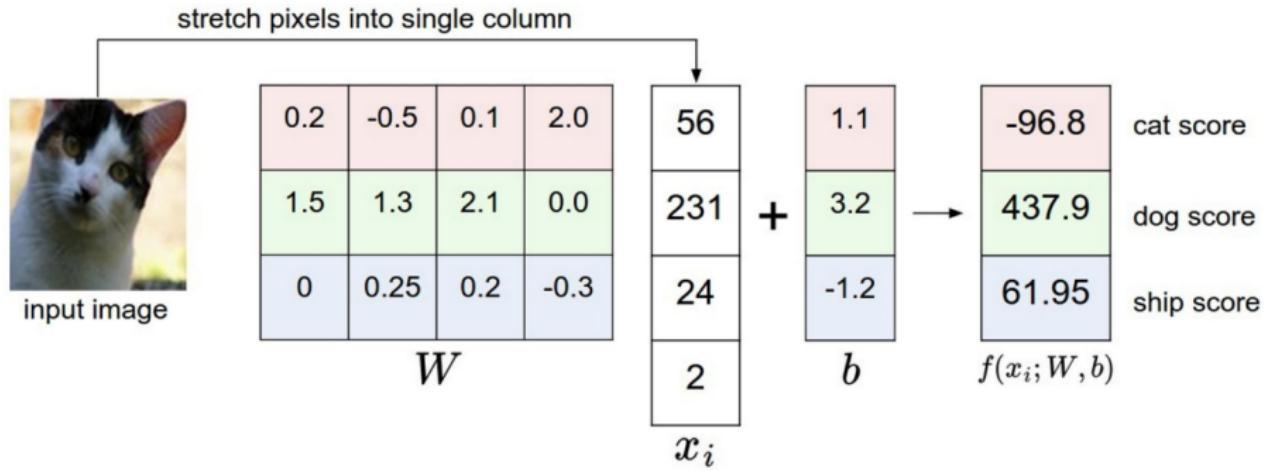
**3072x1**

**(+b)** **10x1**

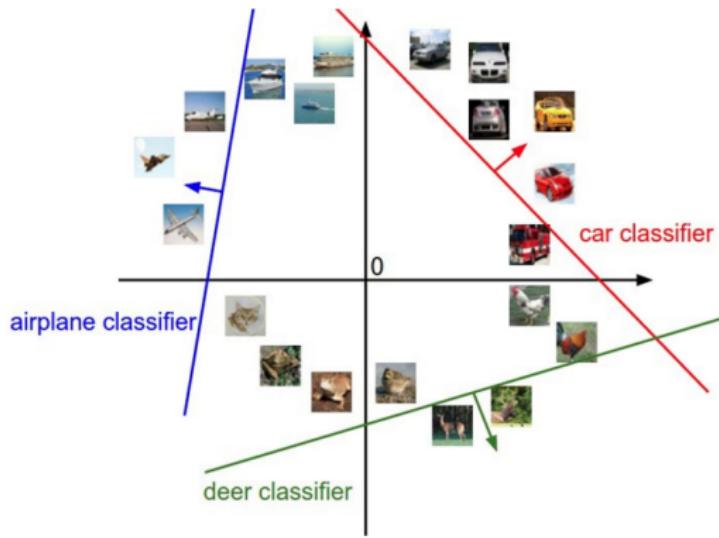
**10** numbers,  
indicating class  
scores

parameters, or “weights”

# Linear classifier: image classification



# Interpreting a Linear Classifier



$$f(x_i, W, b) = Wx_i + b$$



**[32x32x3]**

array of numbers 0...1  
(3072 numbers total)

# Review: The learning problem

## Classification



- **Hypothesis class:** we consider some **restricted** set  $\mathcal{F}$  of mappings  $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{L}$  from images to labels
- **Estimation:** on the basis of a training set of examples and labels,  $\{(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)\}$  we find an estimate  $f \in \mathcal{F}$
- **Evaluation:** we measure how well  $f$  to yet unseen examples i.e whether  $f(x_{new})$  agrees with  $y_{new}$

## Estimation criterion

- We can formulate the binary classification problem more explicitly by defining a zero-one loss:

$$\text{Loss}(y, \hat{y}) = \begin{cases} 0, & y = \hat{y} \\ 1, & y \neq \hat{y} \end{cases} \quad (2)$$

$$\frac{1}{n} \sum_{i=1}^n \text{Loss}(y_i, \hat{y}_i) = \frac{1}{n} \sum_{i=1}^n \text{Loss}(y_i, f(\mathbf{x}_i; \theta)) \quad (3)$$

give the fractions of prediction errors on the training set

- This is a function of the parameters  $\theta$  and we can try to minimize it directly

Suppose: 3 training examples, 3 classes. With some  $W$  the scores are:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

## Multiclass SVM loss:

- Given an example  $\{(x_i, y_i)\}$  where  $x_i$  is the image and where  $y_i$  is the (integer) label, and using the shorthand for the scores vector:  $s_i = f(x_i, W)$

the SVM loss has the form:

$$\mathcal{L}_i = \sum_{i \neq j} \max(0, s_j - s_{y_i} + 1)$$

Suppose: 3 training examples, 3 classes. With some  $W$  the scores are:

Suppose: 3 training examples, 3 classes.  
With some  $W$  the scores  $f(x, W) = Wx$  are:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:			<b>2.9</b>

## Multiclass SVM loss:

- Given an example  $\{(x_i, y_i)\}$  where  $x_i$  is the image and where  $y_i$  is the (integer) label, and using the shorthand for the scores vector:  $s_i = f(x_i, W)$

the SVM loss has the form:

$$\mathcal{L}_i = \sum_{i \neq j} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned}
 &= \max(0, 5.1 - 3.2 + 1) \\
 &+ \max(0, -1.7 - 3.2 + 1) \\
 &= 2.9
 \end{aligned}$$

Suppose: 3 training examples, 3 classes. With some  $W$  the scores are:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	2.9	<b>0</b>	

## Multiclass SVM loss:

- Given an example  $\{(x_i, y_i)\}$  where  $x_i$  is the image and where  $y_i$  is the (integer) label, and using the shorthand for the scores vector:  $s_i = f(x_i, W)$

the SVM loss has the form:

$$\mathcal{L}_i = \sum_{i \neq j} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned}
 &= \max(0, 1.3 - 4.9 + 1) \\
 &+ \max(0, 2.0 - 4.9 + 1) \\
 &= 0
 \end{aligned}$$

Suppose: 3 training examples, 3 classes. With some  $W$  the scores are:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	2.9	0	<b>10.9</b>

## Multiclass SVM loss:

- Given an example  $\{(x_i, y_i)\}$  where  $x_i$  is the image and where  $y_i$  is the (integer) label, and using the shorthand for the scores vector:  $s_i = f(x_i, W)$

the SVM loss has the form:

$$\mathcal{L}_i = \sum_{i \neq j} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned}
 &= \max(0, 2.2 - (-3.1) + 1) \\
 &+ \max(0, 2.5 - (-3.1) + 1) \\
 &= 10.9
 \end{aligned}$$

Suppose: 3 training examples, 3 classes. With some  $W$  the scores are:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	2.9	0	10.9

## Multiclass SVM loss:

- Given an example  $\{(x_i, y_i)\}$  where  $x_i$  is the image and where  $y_i$  is the (integer) label, and using the shorthand for the scores vector:  $s_i = f(x_i, W)$

the SVM loss has the form:

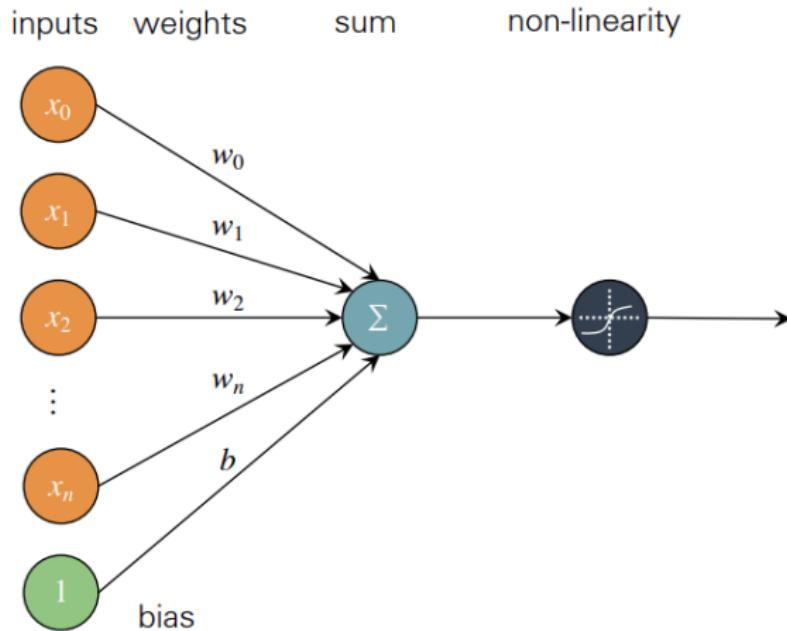
$$\mathcal{L}_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad (4)$$

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \mathcal{L}_i$$

$$\mathbf{L} = (2.9 + 0 + 10.9)/3$$

$$= 4.6$$

# The Perceptron



# Perceptron

- Neuron preactivations (or input activation)

$$a(x) = b + \sum_i w_i x_i = w^T X$$

- Neuron output activation:

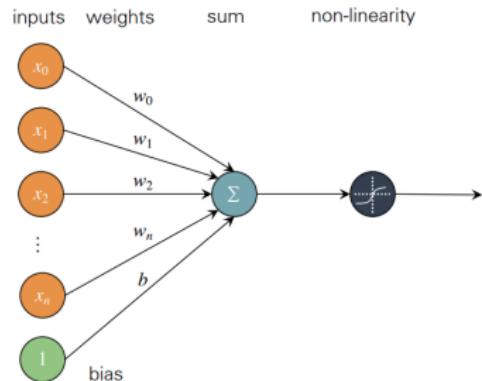
$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

where

$w$  are the weights (parameters)

$b$  is ht bias term

$g(\cdot)$  is the activation function



# Output Activation of The Neuron

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

Range is determined by  $g(\cdot)$

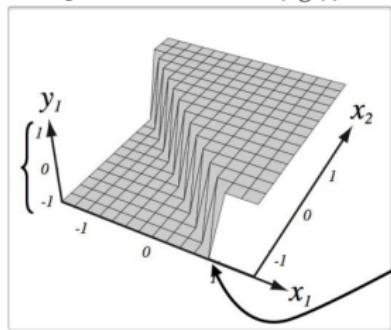
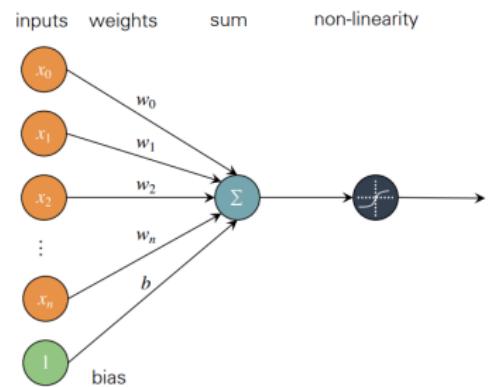


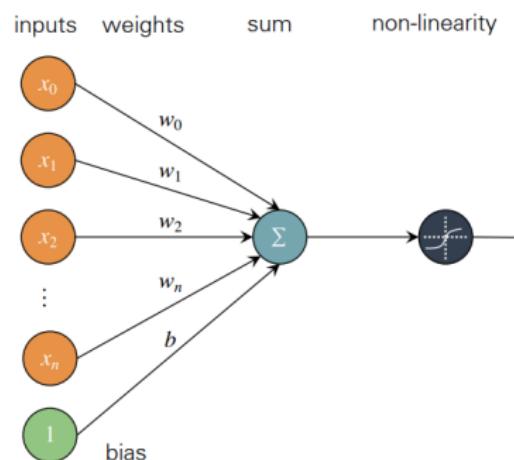
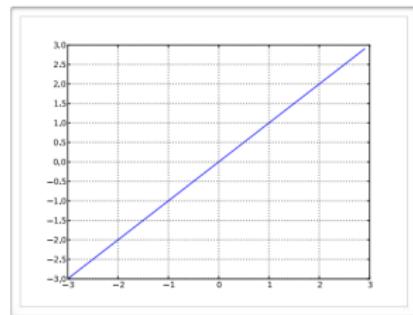
Image credit: Pascal Vincent



# Linear Activation Function

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

$$g(a) = a$$

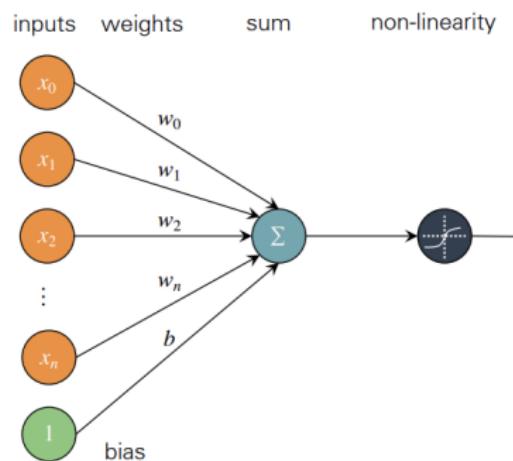
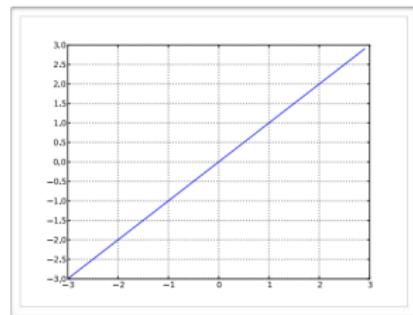


- No non-linear transform
- No squashing

# Linear Activation Function

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

$$g(a) = a$$

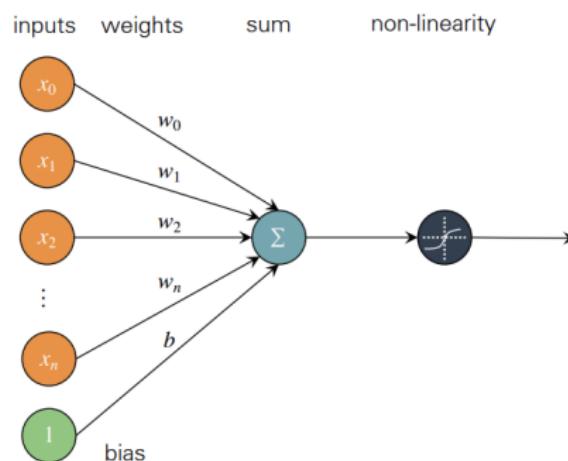
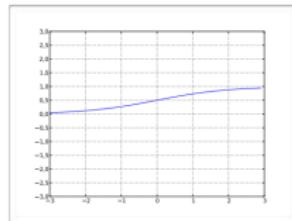


- No non-linear transform
- No squashing

# Sigmoid Activation Function

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

$$g(a) = \text{sigm}(a) = \frac{1}{1 + \exp(-a)}$$

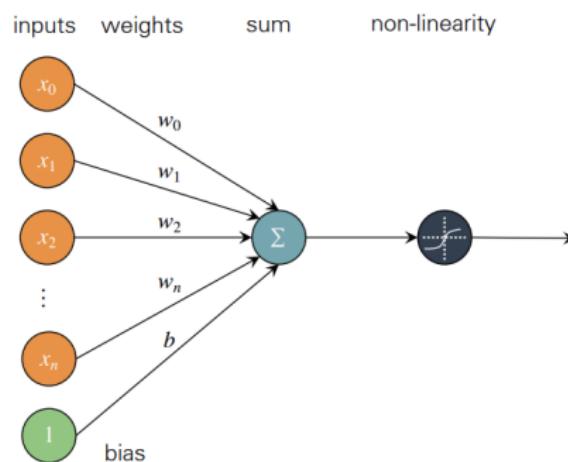
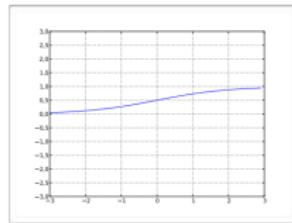


- Squash neuron output between 0 and 1
- Always positive
- Bounded

# Sigmoid Activation Function

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

$$g(a) = \text{sigm}(a) = \frac{1}{1 + \exp(-a)}$$

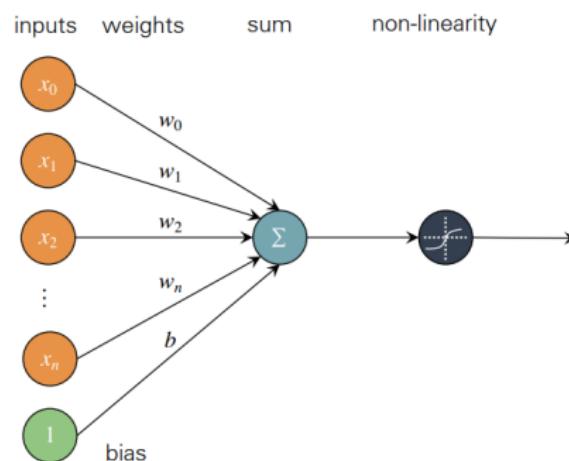
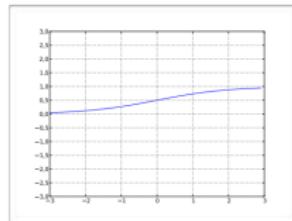


- Squash neuron output between 0 and 1
- Always positive
- Bounded

# Sigmoid Activation Function

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

$$g(a) = \text{sigm}(a) = \frac{1}{1 + \exp(-a)}$$



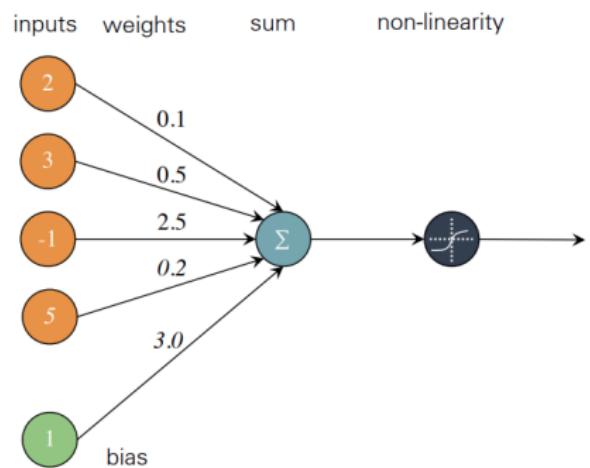
- Squash neuron output between 0 and 1
- Always positive
- Bounded

# Perceptron Forward Pass

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

$$h(x) = g(3.2) = \sigma(3.2)$$

$$\frac{1}{1 + e^{-3.2}} = 0.96$$

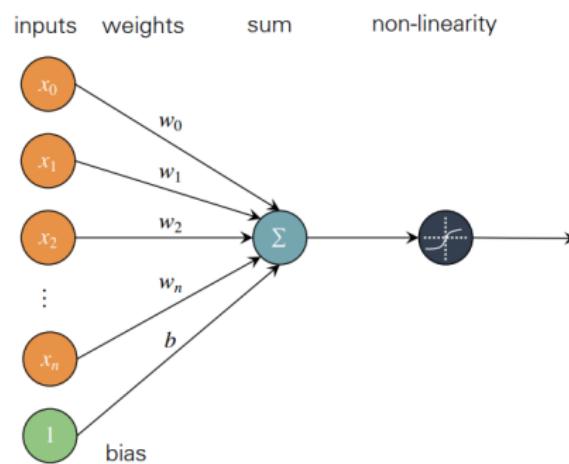
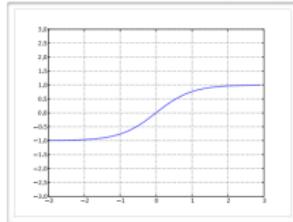


# Hyperbolic Tangent (tanh) Activation Function

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

$$g(a) = \tanh(a) =$$

$$= \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)} = \frac{\exp(2a) - 1}{\exp(2a) + 1}$$



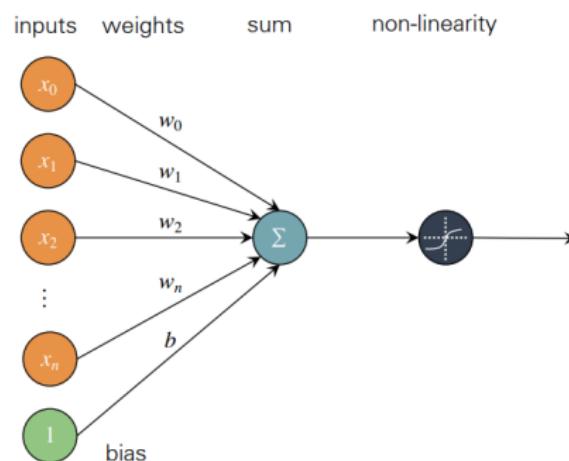
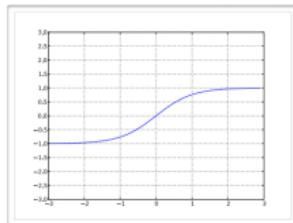
- Squashes the neuron output between 1 and -1
- Bounded

# Hyperbolic Tangent (tanh) Activation Function

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

$$g(a) = \tanh(a) =$$

$$= \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)} = \frac{\exp(2a) - 1}{\exp(2a) + 1}$$

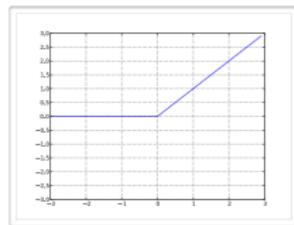


- Squashes the neuron output between 1 and -1
- Bounded

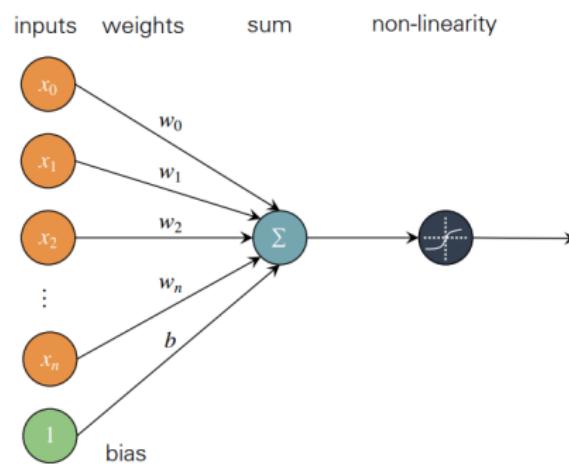
# Rectified Linear (ReLU) Activation Function

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

$$g(a) = \text{Relu}(a) == \max(0, a)$$



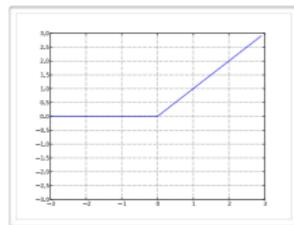
- Bounded by 0
- Not upper bound



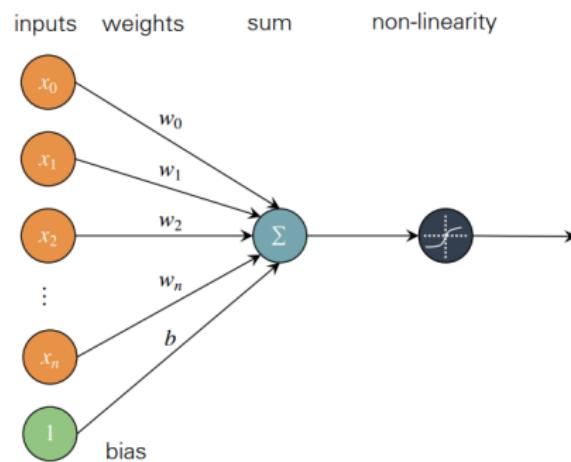
# Rectified Linear (ReLU) Activation Function

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

$$g(a) = \text{Relu}(a) == \max(0, a)$$

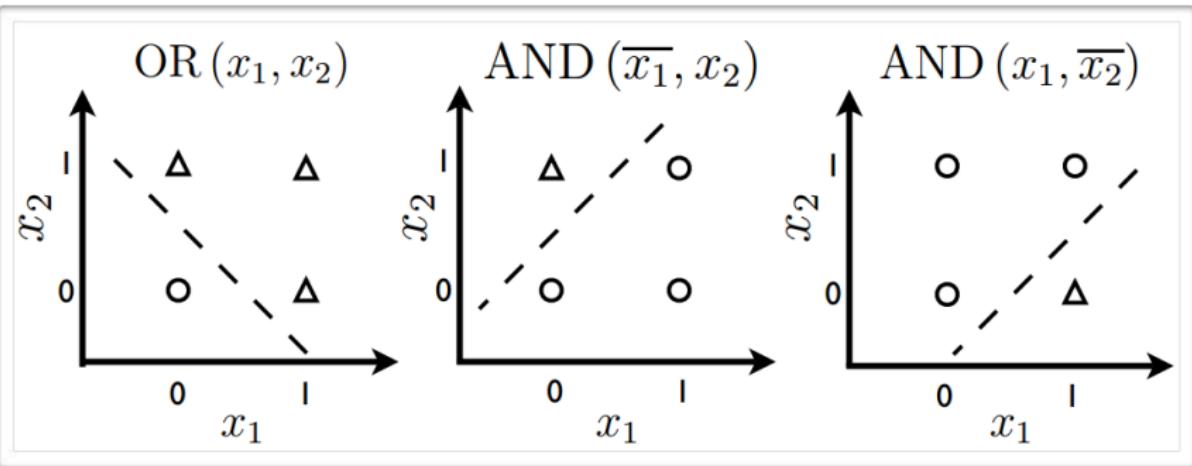


- Bounded by 0
- Not upper bound



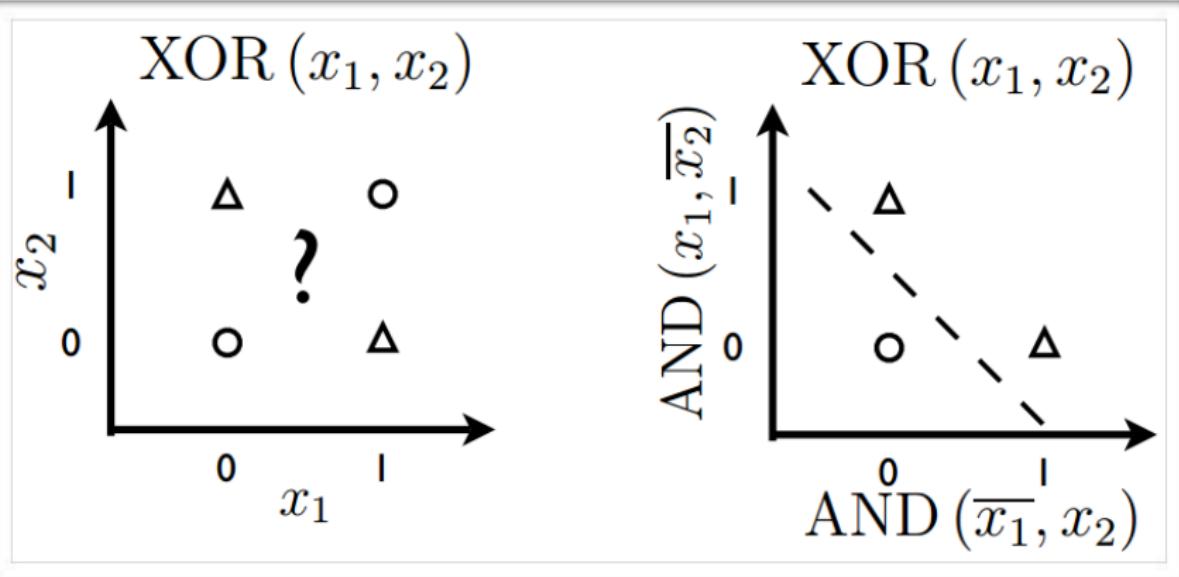
# Capacity of Single Neuron

- Can solve linearly separable problems



# Capacity of Single Neuron

- Can not solve non-linearly separable problems

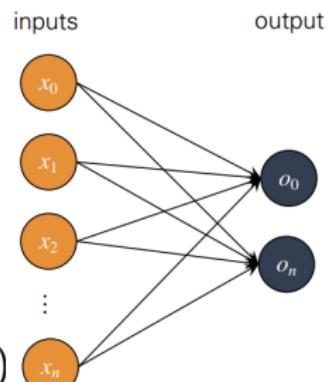


- Need to transform the input into a better representation
- Remember the Basis Functions!!

# Multi-Output Perceptron

- Remember multi-way classification
  - We need multiple outputs (1 output per class)
  - We need to estimate conditional probability  $p(y = c | x)$
  - Discriminative Learning
- Softmax activation function at the output

$$\mathbf{o}(\mathbf{a}) = \text{softmax}(\mathbf{a}) = \left[ \frac{\exp(a_1)}{\sum_c \exp(a_c)} \cdots \frac{\exp(a_C)}{\sum_c \exp(a_c)} \right]$$



- strictly positive
- sums to one
- Predict class with the highest estimated class conditional probability

# Multi-Layer Perceptron

Multi-Layer Perceptron

# Single Hidden Layer Neural Network

- Hidden layer pre-activation

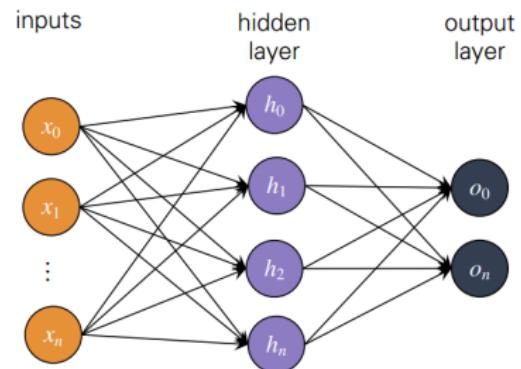
$$a(x) = b^{(1)} + W^{(1)}x$$

- Hidden layer activation:

$$h(x) = g(a(x))$$

- Output layer activation

$$o(x) = o(b^{(2)} + w^{(2)}h^{(1)}x)$$



# Multi-Layer Perceptron (MLP)

- Consider a network with L hidden layers

- layer pre-activation for  $k > 1$

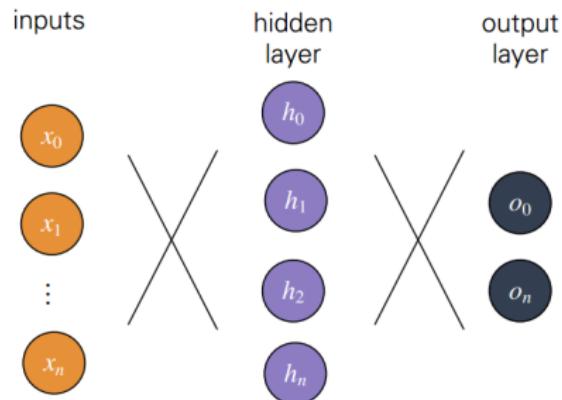
$$a^{(k)}(x) = b^{(k)} + W^{(k)} h^{(k-1)}(x)$$

- hidden layer activation from 1 to L

$$h^{(k)}(x) = g(a^{(k)}(x))$$

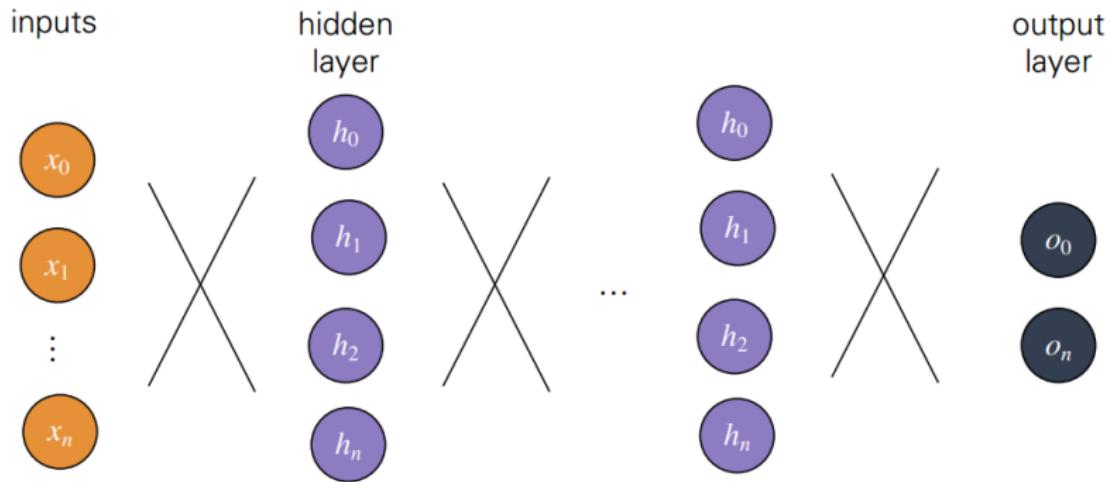
- output layer activation ( $k=L+1$ )

$$h^{(L+1)}(x) = o(a^{(L+1)}(x)) = f(x)$$



$$(\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x})$$

# Deep Neural Network



# Deep Learning Training

Next Class..